GTSuite Client

Technical Documentation

Ionic/Angular Frontend with Metadata-Driven UI

GTsoftware di Giancarlo Thiella

Generated: November 20, 2025

Table of Contents

1. Executive Summary

GTSuite Client is a sophisticated Ionic/Angular hybrid mobile application that serves as the frontend interface for the GTSuite metadata-driven application framework. Built using Angular 20 and Ionic 8, it provides a responsive, cross-platform experience that works seamlessly on web, iOS, and Android devices.

Key Features

- **Metadata-Driven UI:** Entire user interface dynamically rendered from server metadata
- **Cross-Platform:** Single codebase for web, iOS, and Android via Ionic and Capacitor
- **Modern Stack:** Angular 20 with standalone components and signals
- **DevExtreme Integration:** Enterprise-grade UI components for data grids and forms
- **Dynamic Routing:** Component registry system for runtime route configuration
- **Offline Capability:** Ionic Storage for local data persistence
- **Real-time Updates:** Socket.io integration for live notifications
- **Advanced Components:** Reusable GTS framework components (forms, grids, toolbars)
- **AI Integration:** Built-in support for ChatGPT, Claude, and Gemini
- **Comprehensive Admin:** GTSW module for complete portal administration
- **Security:** JWT authentication with OAuth2 (Google, Microsoft) and 2FA support
- **Internationalization:** Multi-language support with dynamic translation

2. Architecture Overview

Application Architecture

The GTSuite client follows Angular's modular architecture with a clear separation between core framework components and feature-specific modules. The metadata-driven approach allows the application to dynamically adapt its UI without requiring code deployments.

Architectural Layers

Layer	Component	Description
Presentation	Ionic Components	UI components and pages
Framework	Core GTS Components	Reusable forms, grids, toolbars
Services	Angular Services	Business logic and API communication
State	RxJS & Observables	Reactive state management
Storage	Ionic Storage	Local persistence layer
Communication	HTTP & WebSockets	Server communication

Project Structure

```
/src
+-- app/
|   +-- core/               # Framework core
|   |   +-- gts/           # GTS components
|   |   +-- services/      # Core services
|   |   +-- guards/        # Route guards
|   |   +-- interceptors/  # HTTP interceptors
|   |   +-- models/        # Data models
|   |   +-- config/        # Configuration
|   +-- features/          # Feature modules
|       +-- auth/          # Authentication
|       +-- GTSW/          # Admin module
|       +-- home/          # Home page
|       +-- profile/       # User profile
|       +-- shell/         # App shell
+-- assets/                # Static resources
+-- environments/          # Environment configs
+-- theme/                 # Styling
```

3. Technology Stack

Core Technologies

Technology	Version	Purpose
Angular	20.0.0	Frontend framework
Ionic	8.0.0	Hybrid mobile framework
Capacitor	7.4.4	Native device capabilities
TypeScript	5.8.0	Type-safe development
RxJS	7.8.0	Reactive programming
DevExtreme	25.1.6	Enterprise UI components

Key Libraries & Tools

- **Ionic Storage:** Local data persistence with IndexedDB
- **ExcelJS:** Excel file generation and manipulation
- **File-Saver:** Client-side file download capabilities
- **CryptoJS:** Encryption and security utilities
- **PDF Viewer:** ngx-extended-pdf-viewer for document display
- **Ionicons:** Icon library for Ionic components
- **ESLint:** Code quality and style enforcement
- **Karma & Jasmine:** Testing framework

DevExtreme Components Used

The application leverages DevExtreme's enterprise-grade components for advanced data presentation and interaction:

- **DataGrid:** Advanced data table with sorting, filtering, and editing
- **Form:** Dynamic form generation with validation
- **Popup:** Modal dialogs and overlays
- **Lookup:** Searchable dropdown components
- **Toolbar:** Action bars and command buttons

- **LoadPanel:** Loading indicators
- **TabPanel:** Tabbed interfaces

4. Metadata-Driven UI System

The GTSuite client implements a sophisticated metadata-driven UI system where the entire user interface is dynamically generated based on metadata retrieved from the server. This approach provides unprecedented flexibility and eliminates the need for client-side code deployments when modifying application structure.

Metadata Flow

1. **Authentication:** User logs in and receives JWT token
2. **Menu Request:** Client requests menu structure from /api/data/getMenuData
3. **Dynamic Routing:** Menu items map to routes via component registry
4. **Page Request:** When user navigates, client calls /api/data/getPageData2
5. **Metadata Processing:** Server returns complete page metadata (forms, grids, actions)
6. **UI Generation:** GTS framework components render UI based on metadata
7. **Action Execution:** User interactions trigger actions defined in metadata
8. **Data Operations:** Actions execute database operations via /api/db endpoints

Component Registry System

The client uses a component registry to map routes to Angular components at runtime. This allows new pages to be added without modifying the routing configuration:

```
export const COMPONENT_REGISTRY = {
  'GTSW/users': () => import('features/GTSW/users'),
  'GTSW/setup': () => import('features/GTSW/setup'),
  // Dynamic entries added at runtime
};
```

Page Metadata Structure

Each page is defined by comprehensive metadata that includes all UI elements and behavior:

- **Page Configuration:** Title, URL, initial action, layout settings
- **Tabs:** Multi-tab layouts with independent content
- **Toolbars:** Action buttons with icons, labels, and behaviors
- **Views:** Data presentation modes (grid, form, detail)
- **Forms:** Field definitions with validation and formatting

- **Grids:** Column definitions, sorting, filtering, editing
- **Actions:** Event handlers with conditional execution rules
- **Data Adapters:** Data source configurations and queries
- **Messages:** User notifications and dialogs
- **Conditional Rules:** Business logic for field visibility and validation

5. Core Framework Components

The GTSuite framework provides a set of reusable Angular components that interpret metadata and render the appropriate UI. These components form the building blocks of the metadata-driven interface.

GTS Components

Component	Purpose
gts-form	Dynamic form generation from metadata
gts-form-popup	Modal form dialogs
gts-grid	Data grid with CRUD operations
gts-toolbar	Action toolbar with buttons
gts-tabs	Tabbed interface container
gts-lookup	Searchable dropdown fields
gts-items	List and detail views
gts-message	User notifications and dialogs
gts-loader	Loading indicators
gts-reports	Report viewer and generator
gts-pdf	PDF document viewer
gts-file-uploader	File upload component
gts-html	HTML content renderer
gts-ai	AI chat interface
gts-debug	Debug information panel
gts-actions-debug	Action execution debugger

Component Features

- **Metadata Interpretation:** Each component reads and interprets metadata specifications
- **Dynamic Rendering:** UI elements created at runtime based on metadata
- **Event Handling:** User interactions trigger metadata-defined actions
- **Validation:** Built-in validation based on metadata rules
- **Responsive Design:** Automatic adaptation to different screen sizes
- **Accessibility:** ARIA labels and keyboard navigation support
- **Performance:** OnPush change detection and lazy loading

- **Reusability:** Components can be nested and composed

GTS Form Component

The gts-form component is one of the most complex framework components, capable of rendering complete forms with multiple field types, validation rules, and conditional logic.

- Text inputs, Text areas, Number inputs
- Date pickers, Time pickers, DateTime pickers
- Dropdowns, Multi-select, Lookup fields
- Checkboxes, Radio buttons, Switches
- File uploaders, Image pickers
- Rich text editors, HTML content
- Custom validators, Async validators

6. Services & State Management

The application uses Angular services for business logic, API communication, and state management. All services follow the singleton pattern and are injected via Angular's dependency injection.

Core Services

Service	Purpose
GtsDataService	Main data service - API communication, action execution, data operations
AuthService	Authentication management, JWT tokens, user sessions, OAuth integration
MenuService	Menu structure management and navigation
PageService	Page metadata caching and retrieval
AppInfoService	Application state and configuration
TranslationService	Internationalization and multi-language support
EncryptionService	Data encryption and security utilities
DynamicRoutesService	Runtime route configuration

GtsDataService

The GtsDataService is the largest and most critical service (2500+ lines), handling all data operations:

- **API Communication:** HTTP requests to all backend endpoints
- **Action Execution:** Interprets and executes metadata-defined actions
- **Data Operations:** CRUD operations on external databases
- **Report Generation:** Report execution and data retrieval
- **File Operations:** Upload, download, and file management
- **Validation:** Client-side data validation
- **Caching:** Page data and metadata caching
- **Error Handling:** Consistent error management and user notification

State Management Pattern

The application uses RxJS observables and subjects for reactive state management:

```
// Service with observable state
private userSubject = new Subject<User>();
public user$ = this.userSubject.asObservable();

// Component subscription
this.authService.user$.subscribe(user => {
  this.currentUser = user;
});
```

7. Authentication & Security

Authentication Flow

1. **Login Page:** User enters credentials or selects OAuth provider
2. **Authentication Request:** Credentials sent to /api/auth/login
3. **Token Receipt:** Server returns JWT token and user data
4. **Token Storage:** Token stored securely in Ionic Storage
5. **HTTP Interceptor:** Token automatically added to all API requests
6. **Route Guard:** Protected routes check authentication status
7. **Token Refresh:** Automatic token refresh before expiration
8. **Logout:** Token cleared and user redirected to login

Authentication Methods

- **Username/Password:** Traditional credential-based login
- **Google OAuth:** Sign in with Google account
- **Microsoft OAuth:** Sign in with Microsoft account
- **Two-Factor Authentication:** Optional TOTP-based 2FA
- **Email Verification:** Account activation via email link
- **Password Reset:** Forgot password flow with email verification

Security Features

- **JWT Tokens:** Stateless authentication with signed tokens
- **HTTPS:** Encrypted communication with backend
- **XSS Protection:** Angular's built-in sanitization
- **CSRF Protection:** Token-based CSRF prevention
- **Route Guards:** Prevent unauthorized access to protected routes
- **HTTP Interceptors:** Automatic token injection and error handling
- **Secure Storage:** Encrypted local storage for sensitive data

- **Session Management:** Automatic logout on token expiration
- **reCAPTCHA:** Bot protection on registration and login

8. GTSW Admin Module

The GTSW (GTSuite Web) module is the administration interface for the GTSuite portal. It provides comprehensive tools for managing users, configuring the system, and monitoring operations. Access to GTSW is restricted to administrators with specific permissions.

GTSW as Reference Implementation

IMPORTANT: The GTSW project serves as the reference implementation and template for all other client projects within the GTSuite framework. It demonstrates the standard architecture, coding patterns, and best practices that should be followed when developing new projects.

- **Standard HTML Templates:** All GTSW page templates follow the same structure and are reused across all projects
- **Common Methods:** Service methods, data handling, and API communication patterns are identical across projects
- **Consistent Architecture:** Component structure, routing, and state management follow the same philosophy
- **Reusable Patterns:** Form validation, grid operations, and action execution use the same implementation
- **Development Guide:** GTSW serves as the live example for onboarding new developers
- **Code Reference:** When developing new projects, GTSW components should be used as templates

Standardized HTML Templates

All pages across different projects (GTR, DCW, GTSW, etc.) use essentially the same HTML template structure. This standardization ensures:

- **Consistency:** Users experience the same interface across all applications
- **Maintainability:** Bugs fixed in one project benefit all projects
- **Rapid Development:** New projects can be scaffolded quickly from existing templates
- **Training:** Developers learn once and apply knowledge everywhere
- **Testing:** Test suites can be reused across projects

```
Typical Page Template Structure:
<ion-header>
  <gts-toolbar [metadata]="toolbarMeta"></gts-toolbar>
</ion-header>

<ion-content>
  <gts-tabs [metadata]="tabsMeta">
    <gts-grid [metadata]="gridMeta"></gts-grid>
    <gts-form [metadata]="formMeta"></gts-form>
  </gts-tabs>
</ion-content>
```

GTSW Components

Component	Purpose
Users	User account management, roles, profiles, and permissions
Auth Rules	Authentication rules and password policies
Auth Mails	Email templates for authentication flows
Granted Objects	Object-level permission management
DB Connections	External database connection configuration
Setup	System configuration and global settings
Logs	System logs viewer and analysis
Languages	Multi-language support and translations
Scheduler	Scheduled task configuration and monitoring
Mail Merge	Email template and merge configuration
Sequences	Auto-increment sequence management
AI Instructions	AI model prompts and configuration

User Management

The Users component provides complete user lifecycle management:

- Create, edit, and delete user accounts
- Assign users to projects and roles
- Manage user profiles and authentication settings
- Configure 2FA and password policies per user
- View user login history and activity logs
- Grant or revoke object-level permissions

- Send password reset and activation emails
- Bulk user import and export

System Configuration

The Setup component allows administrators to configure global system settings:

- Application title, logo, and branding
- Default language and available languages
- Email service configuration (SMTP)
- OAuth provider settings (Google, Microsoft)
- Session timeout and security policies
- File upload limits and allowed types
- Database connection pooling settings
- Backup and maintenance schedules

Monitoring & Logs

The Logs component provides comprehensive system monitoring:

- **Authentication Logs:** Login attempts, OAuth flows, 2FA verification
- **Error Logs:** Application errors, exceptions, and stack traces
- **Database Logs:** Query execution, performance metrics, connection issues
- **SQLite Import Logs:** Metadata import operations and results
- **Report Logs:** Report execution history and parameters
- **Scheduled Task Logs:** Cron job execution and outcomes
- **Email Logs:** Sent emails and delivery status

9. Configuration

The client application is configured through environment files that specify API endpoints, feature flags, and application settings.

Environment Files

- **environment.ts:** Development configuration
- **environment.prod.ts:** Production configuration
- **Angular.json:** Build and deployment settings
- **capacitor.config.ts:** Native app configuration
- **ionic.config.json:** Ionic CLI configuration

Environment Configuration

Setting	Description	Example
apiUrl	Backend API base URL	http://localhost:3000/api
localUrl	Client app URL	http://localhost:8100
reCaptchaEnabled	Enable reCAPTCHA	true
reCaptchaKey	Google reCAPTCHA key	6Lc...
dbMode	Database mode	D, T, P
languageId	Default language	IT, EN
signWithGoogle	Enable Google OAuth	true
signWithMicrosoft	Enable Microsoft OAuth	true
TOTP2FAEnabled	Enable 2FA	true
aiEnabled	Enable AI features	true

10. Installation & Development

Prerequisites

- **Node.js:** Version 18.x or higher
- **npm:** Version 9.x or higher
- **Angular CLI:** Version 20.x (@angular/cli)
- **Ionic CLI:** Version 7.x (@ionic/cli)
- **Git:** For source code management

Installation Steps

Install dependencies:

```
npm install
```

Configure environment:

```
Edit src/environments/environment.ts
```

Start development server:

```
npm start<br/># Or: ionic serve
```

Access application:

```
http://localhost:8100
```

Development Commands

- **npm start:** Start dev server with increased memory
- **npm run build:** Production build
- **npm test:** Run unit tests
- **npm run lint:** Run ESLint code quality checks
- **ionic serve:** Start Ionic dev server
- **ionic build:** Build for web deployment

11. Building & Deployment

Web Deployment

Building the application for web deployment:

```
# Production build
ng build --configuration production

# Output directory
www/

# Deploy to web server
# Copy contents of www/ to web server
```

iOS Deployment

Add iOS platform:

```
ionic capacitor add ios
```

Build web assets:

```
ionic build --prod
```

Copy to iOS:

```
ionic capacitor copy ios
```

Open Xcode:

```
ionic capacitor open ios
```

Build in Xcode:

```
Build and archive in Xcode for App Store
```

Android Deployment

Add Android platform:

```
ionic capacitor add android
```

Build web assets:

```
ionic build --prod
```

Copy to Android:

```
ionic capacitor copy android
```

Open Android Studio:

```
ionic capacitor open android
```

Build APK/AAB:

```
Build signed APK in Android Studio
```

12. Integration with Backend

The client integrates seamlessly with the GTSuite Node.js backend through REST APIs and WebSocket connections. All communication is secured with JWT tokens and HTTPS.

API Integration

- **/api/auth:** Authentication and user management
- **/api/data:** Metadata retrieval and operations
- **/api/db:** External database queries and procedures
- **/api/files:** File upload and download
- **/api/mail:** Email sending services
- **/api/setup:** System configuration
- **/api/task:** Scheduled task management
- **/api/prj:** Project management
- **/api/sqlite:** SQLite metadata import

HTTP Interceptor

The application uses an HTTP interceptor to automatically inject JWT tokens and handle errors:

- Automatic JWT token injection in Authorization header
- Token refresh on 401 Unauthorized responses
- Consistent error handling and user notification
- Request/response logging for debugging
- Retry logic for failed requests
- Network connectivity detection

Real-time Communication

The client can establish WebSocket connections for real-time updates:

- Socket.io integration for bidirectional communication
- Live notifications and alerts

- Real-time data updates without page refresh
- Collaborative editing support
- System status monitoring
- Automatic reconnection on network issues

Conclusion

The GTSuite Client represents a modern, sophisticated approach to building enterprise applications with a metadata-driven architecture. By leveraging Angular and Ionic's powerful capabilities along with DevExtreme's enterprise components, it provides a flexible, maintainable, and user-friendly interface that adapts dynamically to business requirements. The GTSW administration module provides comprehensive tools for managing the entire system, making it suitable for enterprise deployments requiring centralized control and monitoring. For additional support or inquiries, please contact GTsoftware di Giancarlo Thiella.